File Handling Functions

- fopen()

The fopen() function opens a file or URL. If fopen() fails, it returns FALSE and an error on failure. You can hide the error output by adding an '@' in front of the function name.

fopen(filename,mode,include_path,context)

Parameter	Description
filename	Required. Specifies the file or URL to open
mode	Required. Specifies the type of access you require to the file/stream. Possible values: "r" (Read only. Starts at the beginning of the file) "r+" (Read/Write. Starts at the beginning of the file) "w" (Write only. Opens and clears the contents of file; or creates a new file if it doesn't exist) "w+" (Read/Write. Opens and clears the contents of file; or creates a new file if it doesn't exist) "a" (Write only. Opens and writes to the end of the file or creates a new file if it doesn't exist) "a+" (Read/Write. Preserves file content by writing to the end of the file) "x" (Write only. Creates a new file. Returns FALSE and an error if file already exists) "x+" (Read/Write. Creates a new file. Returns FALSE and an error if file already exists)
include_path	Optional. Set this parameter to '1' if you want to search for the file in the include_path (which is set in php.ini) as well
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

Example

```
<?php
$file = fopen("test.txt","r");
$file = fopen("/home/test/test.txt","r");
$file = fopen("/home/test/test.gif","wb");
$file = fopen("http://www.example.com/","r");
$file = fopen("ftp://user:password@example.com/test.txt","w");
?>
```

- fread()

The fread() reads from an open file. The function will stop at the end of the file or when it reaches the specified length, whichever comes first. This function returns the read string, or FALSE on failure.

fread(file,length)

Parameter	Description
file	Required. Specifies the open file to read from
length	Required. Specifies the maximum number of bytes to read

Example 1

Read 10 bytes from file:

```
<?php
$file = fopen("test.txt","r");
fread($file,"10");
fclose($file);
?>
```

Example 2

Read entire file:

```
<?php
$file = fopen("test.txt","r");
fread($file,filesize("test.txt"));
fclose($file);
?>
```

- fwrite()

The fwrite() writes to an open file. The function will stop at the end of the file or when it reaches the specified length, whichever comes first. This function returns the number of bytes written, or FALSE on failure.

fwrite(file,string,length)

Parameter	Description
file	Required. Specifies the open file to write to
string	Required. Specifies the string to write to the open file
length	Optional. Specifies the maximum number of bytes to write

Example

```
<?php
$file = fopen("test.txt","w");
echo fwrite($file,"Hello World. Testing!");
```

```
fclose($file);
?>
```

The output of the code above will be:

21

- fclose()

The fclose() function closes an open file. This function returns TRUE on success or FALSE on failure.

fclose(file)

Parameter	Description
file	Required. Specifies the file to close

Example

```
<?php
$file = fopen("test.txt","r");
//some code to be executed
fclose($file);
?>
```

- file_exists()

The file_exists() function checks whether or not a file or directory exists. This function returns TRUE if the file or directory exists, otherwise it returns FALSE.

file_exists(path)

Parameter	Description
path	Required. Specifies the path to check

Example

```
<?php
echo file_exists("test.txt");
?>
```

The output of the code above will be:

1

- is_readable()

The is_readable() function checks whether the specified file is readable. This function returns TRUE if the file is readable.

is_readable(file)

Parameter	Description
File	Required. Specifies the file to check

Example

```
<?php
$file = "test.txt";
if(is_readable($file))
  {
  echo ("$file is readable");
  }
else
  {
  echo ("$file is not readable");
  }
?>
```

The output of the code above could be:

test.txt is readable

- is_writeable()

The is_writeable() function checks whether the specified file is writeable. This function returns TRUE if the file is writeable. This function is an alias of the is_writable() function.

is_writeable(file)

Parameter	Description
file	Required. Specifies the file to check

Example

```
<?php
$file = "test.txt";
if(is_writeable($file))
  {
  echo ("$file is writeable");
  }
else
  {
  echo ("$file is not writeable");
  }
?>
```

The output of the code above could be:

test.txt is writeable

- fgetc()

The fgetc() function returns a single character from an open file.

fgetc(file)

Parameter	Description
File	Required. Specifies the file to check

Note: This function is slow and should not be used on large files. If you need to read one character at a time from a large file, use fgets() to read data one line at a time and then process the line one character at a time with fgetc().

Example 1

```
<?php
$file = fopen("test2.txt","r");
echo fgetc($file);
fclose($file);
?>
```

The output of the code above will be: H

Example 2

Read file character by character:

```
<?php
$file = fopen("test2.txt","r");
while (! feof ($file))
  {
  echo fgetc($file);
  }
fclose($file);
?>
```

The output of the code above will be:

Hello, this is a test file.

- fgets()

The fgets() function returns a line from an open file. The fgets() function stops returning on a new line, at the specified length, or at EOF, whichever comes first. This function returns FALSE on failure.

fgets(file,length)

Parameter	Description
File	Required. Specifies the file to read from
length	Optional. Specifies the number of bytes to read. Default is 1024 bytes.

Example 1

```
<?php
$file = fopen("test.txt","r");
echo fgets($file);
fclose($file);
?>
```

The output of the code above will be:

Hello, this is a test file.

Example 2

Read file line by line:

```
<?php
$file = fopen("test.txt","r");
while(! feof($file))
  {
  echo fgets($file). "<br />";
  }
fclose($file);
?>
```

The output of the code above will be:

Hello, this is a test file.
There are three lines here.
This is the last line.

- file()

The file() reads a file into an array. Each array element contains a line from the file, with newline still attached.

file(path,include_path,context)

Parameter	Description
path	Required. Specifies the file to read
include_path	Optional. Set this parameter to '1' if you want to search for the file in the include_path (which is set in php.ini) as well
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream. Can be skipped by using NULL.

Example

```
<?php
print_r(file("test.txt"));
?>
```

The output of the code above will be:

```
Array
(
[0] => Hello World. Testing testing!
[1] => Another day, another line.
[2] => If the array picks up this line,
[3] => then is it a pickup line?
)
```

- file_get_contents()

The file_get_contents() reads a file into a string. This function is the preferred way to read the contents of a file into a string. Because it will use memory mapping techniques, if this is supported by the server, to enhance performance.

file_get_contents(path,include_path,context,start,max_length)

Parameter	Description
path	Required. Specifies the file to read
include_path	Optional. Set this parameter to '1' if you want to search for the file in the include_path (in php.ini) as well
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream. Can be skipped by using NULL.
start	Optional. Specifies where in the file to start reading. This parameter was added in PHP 5.1
max_length	Optional. Specifies how many bytes to read. This parameter was added in PHP 5.1

Example

```
<?php
echo file_get_contents("test.txt");
?>
```

The output of the code above will be:

This is a test file with test text.

Example 2

```
<?php
$homepage = file_get_contents('http://www.google.com/');
echo $homepage;
?>
```

Output :- it will display the content from google home page.

- File_put_contents()

The file_put_contents() writes a string to a file. This function follows these rules when accessing a file:

- If FILE_USE_INCLUDE_PATH is set, check the include path for a copy of *filename*
- Create the file if it does not exist
- Open the file
- Lock the file if LOCK_EX is set
- If FILE_APPEND is set, move to the end of the file. Otherwise, clear the file content
- Write the data into the file
- Close the file and release any locks

This function returns the number of character written into the file on success, or FALSE on failure.

file_put_contents(file,data,mode,context)

Parameter	Description
File	Required. Specifies the file to write to. If the file does not exist, this function will create one
data	Required. The data to write to the file. Can be a string, an array or a data stream
mode	Optional. Specifies how to open/write to the file. Possible values: FILE_USE_INCLUDE_PATH FILE_APPEND LOCK_EX
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream.

Note: Use FILE_APPEND to avoid deleting the existing content of the file.

Example

```
<?php
echo file_put_contents("test.txt","Hello World. Testing!");
?>
```

The output of the code above will be:

21

- ftell()

The ftell() function returns the current position in an open file. Returns the current file pointer position, or FALSE on failure.

ftell(file)

Parameter	Description
File	Required. Specifies the open file to check

Example

```
<?php
$file = fopen("test.txt","r");

// print current position
echo ftell($file);

// change current position
fseek($file,"15");

// print current position again
echo "<br />" . ftell($file);

fclose($file);
?>
```

The output of the code above will be:

```
0
15
```

- fseek()

The fseek() function seeks in an open file. This function moves the file pointer from its current position to a new position, forward or backward, specified by the number of bytes.

This function returns 0 on success, or -1 on failure. Seeking past EOF will not generate an error.

fseek(file,offset,whence)

Parameter	Description
File	Required. Specifies the open file to seek in
offset	Required. Specifies the new position (measured in bytes from the beginning of the file)
whence	Optional. (added in PHP 4). Possible values: SEEK_SET - Set position equal to offset. Default SEEK_CUR - Set position to current location plus offset SEEK_END - Set position to EOF plus offset (to move to a position before EOF, the offset must be a negative value)

Tip: Find the current position by using ftell()!

Example

```
<?php
$file = fopen("test.txt","r");
// read first line
fgets($file);
// move back to beginning of file
fseek($file,0);
?>
```

- rewind()

The rewind() function "rewinds" the position of the file pointer to the beginning of the file. This function returns TRUE on success, or FALSE on failure.

rewind(file)

Parameter	Description
file	Required. Specifies the open file

Example

```
<?php
$file = fopen("test.txt","r");

//Change position of file pointer
fseek($file,"15");
//Set file pointer to 0
rewind($file);
fclose($file);
?>
```

- copy()

The copy() function copies a file. This function returns TRUE on success and FALSE on failure.

copy(file,to_file)

Parameter	Description
File	Required. Specifies the file to copy
to_file	Required. Specifies the file to copy to

Note: If the destination file already exists, it will be overwritten.

Example

```
<?php
echo copy("source.txt","target.txt");
?>
```

The output of the code above will be:

1

- unlink()

The unlink() function deletes a file. This function returns TRUE on success, or FALSE on failure.

unlink(filename,context)

Parameter	Description
filename	Required. Specifies the file to delete
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

Example

```
<?php
$file = "test.txt";
if (!unlink($file))
  {
  echo ("Error deleting $file");
  }
else
  {
  echo ("Deleted $file");
  }
?>
```

- rename()

The rename() function renames a file or directory. This function returns TRUE on success, or FALSE on failure.

rename(oldname,newname,context)

Parameter	Description
oldname	Required. Specifies the file or directory to be renamed
newname	Required. Specifies the new name of the file or directory
context	Optional. Specifies the context of the file handle. Context is a set of options that can modify the behavior of a stream

Example

```
<?php
rename("images","pictures");
?>
```

- move_uploaded_file()

The move_uploaded_file() function moves an uploaded file to a new location. This function returns TRUE on success, or FALSE on failure.

move_uploaded_file(file,newloc)

Parameter	Description
File	Required. Specifies the file to be moved
Newloc	Required. Specifies the new location for the file

Note: This function only works on files uploaded via HTTP POST.
Note: If the destination file already exists, it will be overwritten.

Upload.html

```
<html>
<title>Upload Demo</title>
<form enctype="multipart/form-data" method="post" action="upload2.php">
   Send this file: <input name="userfile" type="file" /><br />
   <input type="submit" value="Send File" />
</form>
</html>
```

Upload2.php

```
<?php
$fileName = $_FILES["userfile"]["name"];
$fileTmpLoc = $_FILES["userfile"]["tmp_name"];
$pathAndName = "uploads/".$fileName; //create this folder in your dir.
$moveResult = move_uploaded_file($fileTmpLoc, $pathAndName);
if ($moveResult == true)
{
   echo "File has been moved from " . $fileTmpLoc . " to " . $pathAndName;
}
else
{
    echo "ERROR: File not moved correctly";
}
?>
```